**Zendrive Shake getsure detection problem**

**Objective:** Create a classifier that can be implemented on streaming sensor data to detect Shake gesture

**Data set description:**

Data set had two types of data:

1. **Sensor data** from various phone sensors, with the following fields:
   1. timestamp(ms)  : Unix Timestamp in milliseconds.
   2. acceleration\_x(g) : Acceleration along phone's x axis (measured in g)
   3. acceleration\_y(g) : Acceleration along phone's y axis (measured in g)
   4. acceleration\_z(g) : Acceleration along phone's z axis (measured in g)
   5. roll(rad) : Roll angle representing phone's attitude (orientation)
   6. pitch(rad) : Pitch angle representing phone's attitude (orientation)
   7. yaw(rad) : Yaw angle representing phone's attitude (orientation)
   8. angular\_velocity\_x(rad/sec) : Angular velocity around x axis (passing thru phones center along smaller edge)
   9. angular\_velocity\_y(rad/sec) : Angular velocity around y axis (passing thru phones center along longer edge)
   10. angular\_velocity\_z(rad/sec) : Angular velocity around z axis (passing thru phones center emerging out of phone )

The recorded data is at an interval of 100ms, which is the raw feature set of the classification model.

1. **Label data**, with the following fields:
   1. timestamp(ms) : Unix Timestamp in milliseconds
   2. Lables : Three levels with “0” as start, “1” end and “2” as cancel (distinct count of the label data showed it had only 0 and 1 as labels)

**Methodology steps:**

1. **Pre-processing of raw data:**
   1. Data importing: Three files each for sensor and label data are imported and merged for further processing
   2. Data labeling : A new column named ‘shake’ is added to the dataframe that represents the shake status. This column is filled from the label data. All rows placed between label ‘0’ and label ‘1’ are filled with ‘1’ and viceversa. The occurrence of shake are labeled as ‘1’ and non-shake condition are labeled as ‘0’.

The following is the distribution of Events(Shake) and Non-events(No-shake) in the entire dataset:

|  |  |  |
| --- | --- | --- |
| **Events (1)** | **Non-events (0)** | **Events (%)** |
| 2548 | 163868 | 1.56% |

1. **Feature generation:**

Few features are generated based on intuition like:

1. “derived1” – resultant acceleration = sqrt(acceleration\_x(g)^2 + acceleration\_y(g)^2 + acceleration\_z(g)^2)
2. “derived2” – resultant velocity = sqrt(angular\_velocity\_x(rad/sec)^2 + angular\_velocity\_y(rad/sec)^2 + angular\_velocity\_z(rad/sec)^2)

The assumption behind taking these relative acceleration and velocity is to capture the impact of movement in all three dimensions (which will be the case for shaking) rather than in only one direction

1. churn of all variables ie. Successive difference between two timestamps, with the rational that higher the difference between two successive instance, better the chance to be in shake mode
2. **Pairwise correlation checking for multi-collinearity check:**

Pairwise correlation threshold was taken as 0.75 ie. if two variables has high correlation, one of them was dropped from the model. In this case, there was one such pairs:

|  |  |  |
| --- | --- | --- |
| **Variable 1** | **Variable 1** | **Correlation** |
| angular\_velocity\_x(rad/sec) | angular\_velocity\_y(rad/sec) | 1.00 |

Of these two variables, angular\_velocity\_y(rad/sec) was dropped from the model.

1. **Splitting data into train and test data for cross-validation:**

The entire dataset post variable derivation is split randomly into train and test data set in 80:20 ratios. The following are the counts of events and non-events in the two data set:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Events (1)** | **Non-events (0)** | **Events (%)** |
| Train dataset | 2035 | 129059 | 1.58% |
| Test dataset | 513 | 32261 | 1.59% |
| **Total** | 2548 | 163868 | 1.56% |

1. **Running random forest:**

Random forest algorithm was run on the feature dataset with the following parameters:

1. Dataset – acceleration\_x(g), acceleration\_y(g), acceleration\_z(g), angular\_velocity\_x(rad/sec), angular\_velocity\_z(rad/sec), roll(rad), pitch(rad), yaw(rad), derived1, derived2

2. Number of trees is 100

1. **Feature Importance:**

The following array represents feature importance as computed by random forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **acceleration\_x(g)** | **acceleration\_y(g)** | **acceleration\_z(g)** | **angular\_velocity\_x(rad/sec)** | **angular\_velocity\_y(rad/sec)** |
| 0.152 | 0.124 | 0.071 | 0.077 | 0.128 |
| **roll(rad)** | **pitch(rad)** | **yaw(rad)** | **derived1** | **derived2** |
| 0.082 | 0.081 | 0.067 | 0.123 | 0.090 |

1. **Accuracy**

The following is the confusion matrix for the **training dataset**:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual | |
|  |  | 0 | 1 |
| Predicted | 0 | 129059 | 0 |
| 1 | 0 | 2035 |

The following is the confusion matrix for the **test** **dataset**:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual | |
|  |  | 0 | 1 |
| Predicted | 0 | 32210 | 51 |
| 1 | 302 | 211 |

**Performance measures:**

|  |  |  |
| --- | --- | --- |
|  | **Training dataset** | **Test dataset** |
| **Accuracy** | **100%** | **98.92%** |
| **Precision** | **100%** | **41.13%** |
| **Recall** | **100%** | **80.53%** |

**Appendix:**

**Other approaches taken:**

1. **Logistic regression:**

Stepwise logistic regression was run posts normalizing the features. But the significance level of variables was impacted due to multi-collinearity. Post VIF calculation, I took interaction variables which came out to be significant but the overall accuracy was much less than Random Forest model.

1. **KNN Classification:**

KNN classification was done on normalized feature set. On optimizing the misclassification rate, it was found that k=1 gave the best prediction which is definitely due to overfitting. Also the accuracy and precision was less than Random forest model.

1. **Ensembling :**

Used KNN, Logistic regression and Random-forest for ensemble learning. The gestures which were predicted as events ie. shake by logistic regression was a subset of that predicted by Random Forest. On the other hand, KNN was adding false negative, thus reducing accuracy and recall.